Assignment 3

IMU Dead Reckoning

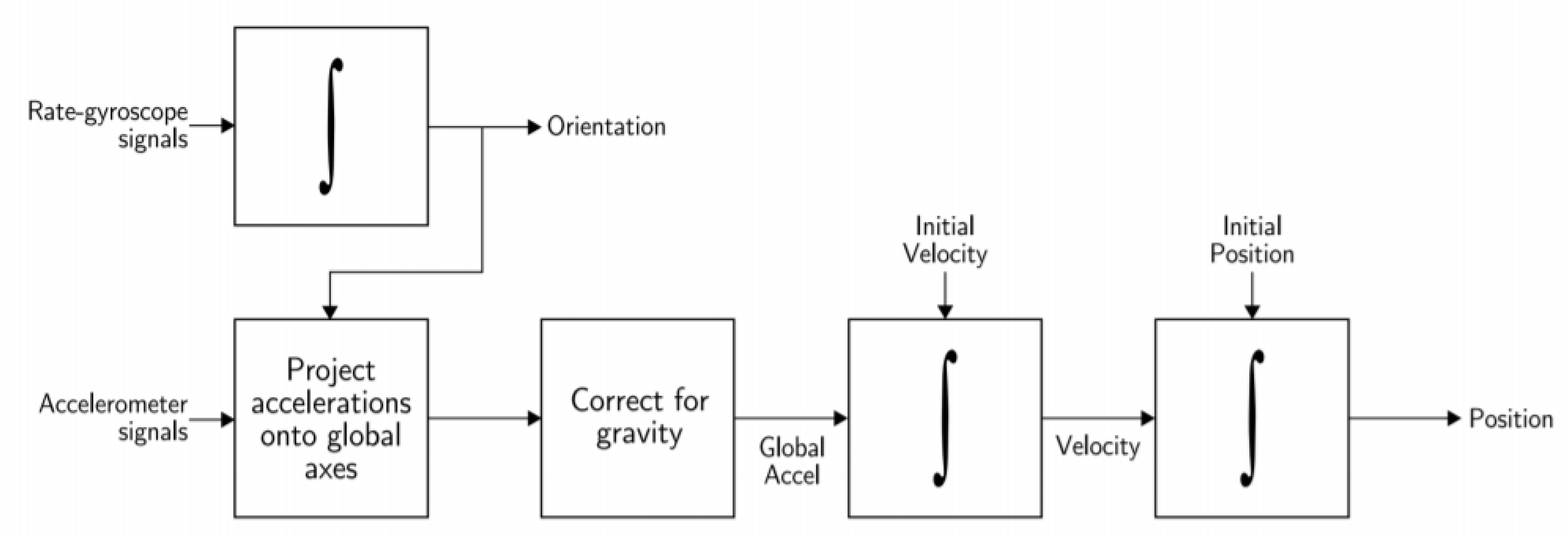
1. Visualize the path of IMU in rviz.

- For sake of simplicity, you only need to do pure IMU integration in this assignment.
- NO filtering is needed!
- If you are not familiar with direction cosine matrices (DCM), a.k.a. rotation matrices, it is advised to go through Part 1 of this article.
- See this paper Chapter 6.1, 6.2 for implementation details.

1. Visualize the path of IMU in rviz.

- In *sdc_hw3.bag*, **/imu/data** is of data type sensor_msgs/Imu.msg
- Please subscribe topic **/imu/data** and use the angular velocity and linear acceleration provided to draw a path.
- To draw the path, you'll need to use LINE_STRIP marker. Please have a look at this document, and understand how to publish the marker.
- If you still don't know how to write subscriber or publisher, please go to ROS wiki and find the information you need.

1. Visualize the path of IMU in rviz.



1. Visualize the path of IMU in rviz.

- We only care about pose relative to initial IMU body frame, so global frame is set to the body frame of the first IMU measurement:

$$s_g(0) = (0,0,0)^T, \qquad C(0) = I$$

- sensor_msgs/Imu:

Header header: Includes timestamp since sampling period may not be regular. Use it to get δt.

geometry_msgs/Quaternion orientation: DO NOT USE!!

geometry_msgs/Vector3 angular_velocity: in $rad/\sec$

geometry_msgs/Vector3 linear_acceleration: in m/s^2

1. Visualize the path of IMU in rviz.

- IMU is initially static in the dataset (or else very difficult to recover gravity vector) and placed on table.
- Assume gravity vector in the global frame (which is the initial IMU body frame in our case) is a constant, which is equal to the first acceleration measurement since the IMU is initially static and only affected by gravity.
- Draw the path of IMU according to the equations in 6.2.2. (this paper)

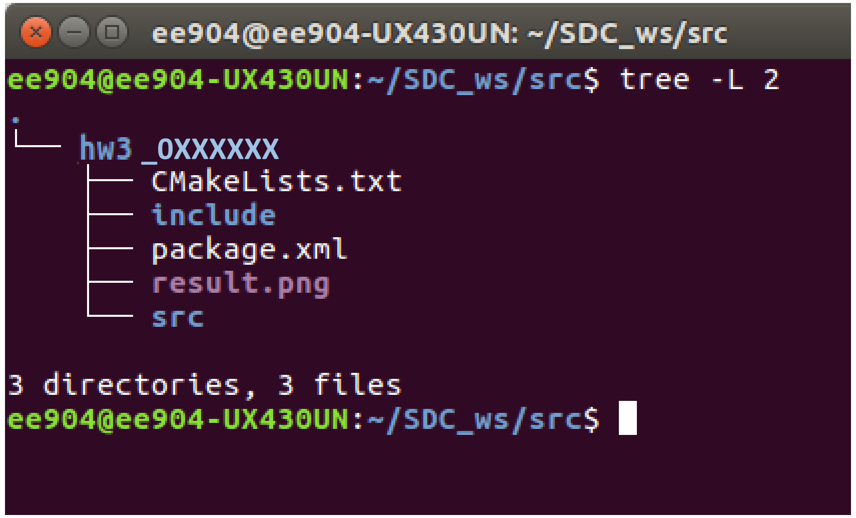
Submission Format

- Your program should publish one **visualization_msgs/marker**, it is the path of IMU (blue).
- You can name the marker topics whatever.
- Name the package **hw3 _<student_id>** and the executable **hw3_node**

Submission Format

- Name your package as **hw3 _<student_id>** and compress your file to **hw3 _<student_id>.zip**(or tar, rar…etc.)
- In the zip file, it should contain

1. Entire package with formulated name
2. The screenshot of the path visualized on rviz

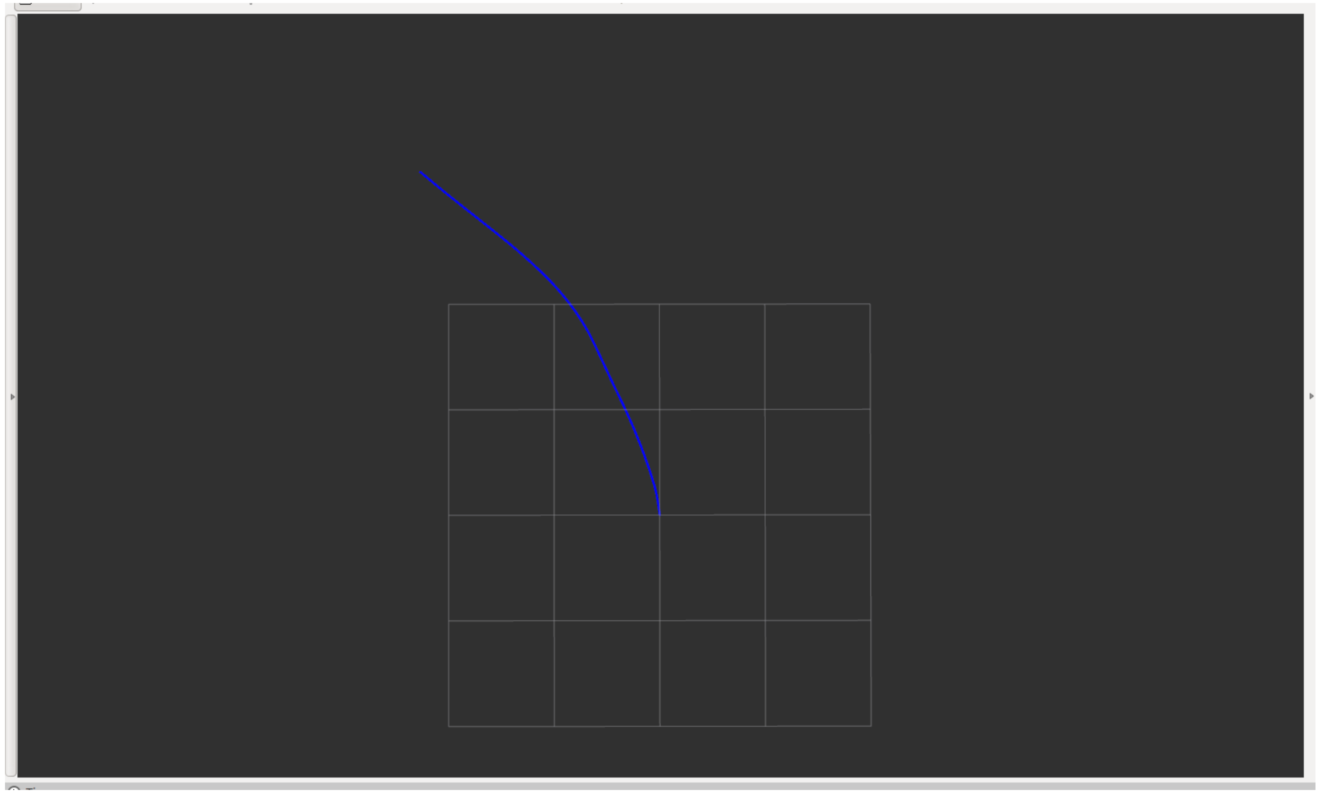


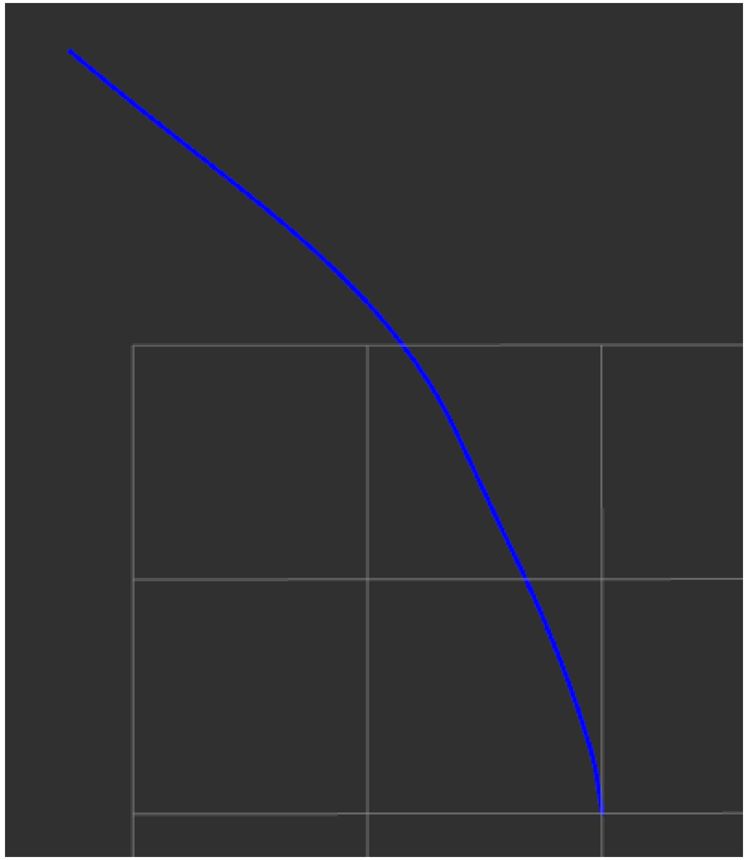
```
ee904@ee904-UX430UN:~/SDC_ws/src$ tree -L 2
.
└── hw3 _0XXXXXX
    ├── CMakeLists.txt
    ├── include
    ├── package.xml
    ├── result.png
    └── src

3 directories, 3 files
ee904@ee904-UX430UN:~/SDC_ws/src$ 
```

Submission Format

- Your screenshot may look like this: *IMU (blue)*





Eigen

- You will need to do matrix operations during implementation : Use **Eigen3**
- Note: It is already installed when installing ROS
- If not, you can install Eigen3:

```
$ sudo apt-get install libeigen3-dev
```

- Quick tutorial
- Eigen is a header only library
  - ➢Only need to include header file: ***#include <Eigen/Dense>***
  - ➢Remember to add the following to your CMakeLists.txt:

```
find_package(Eigen3 REQUIRED)
include_directories(${EIGEN3_INCLUDE_DIRS})
```